C502 Dual-Port Sync Board

User's Manual

Moxa Technologies Co., Ltd.
Tel: +866-2-8665-8535
Fax: +886-2-8665-8536
http://www.moxa.com
e-mail: service@moxa.com.tw

C502 Dual-Port Sync Board User's Manual

Copyright Notice

Trademarks

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.

Moxa provides in this document "as is", without warranty of any kind, either expressed or implied, including, but not limited to, the particular purpose. Moxa may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

Information provided in this manual is intended to be accurate and reliable. However, Moxa Technologies assumes no responsibility for its use, or for any infringements of rights of the fourth parties which may result from its use.

This product could include technical or typographical errors. Changes are periodically made to the information herein; these changes may be incorporated in new editions of the publication.

MOXA Internet Services

Customer's satisfaction is always our number one concern. To ensure customers get the full benefit of our services, Moxa Internet Services have been built for technical support, product inquiry, new driver update, user's manual update, etc.

MOXA Internet services are listed below.

E-mail for technical support

service@moxa.com.tw

FTP site for driver update

ftp://ftp.moxa.com or ftp://ftp.moxa.com.tw

User ID: ftp

Password: your_email_address

World Wide Web (WWW) Site for product info

http://www.moxa.com or http://www.moxa.com.tw

Document Organization

Chapter 1, “C502 Overview”, describes features and specifications for MOXA C502.

Chapter 2, “C502 Hardware Installation”, describes how to install C502 card in your PC.

Chapter 3, “C502 Software Installation”, describes how to install/remove C502 Windows NT driver.

Chapter 4, “API Programming Library”, lists all library functions for MOXA C502 with C/++, VB or Delphi language in Windows NT.

TABLE OF CONTENTS

1
Overview

1.1 Features

MOXA C502 is a high-speed intelligent dual-port control card with synchronous communication modules for PC/AT under Windows NT environment. It is equipped with a RISC CPU, 1Mbytes dual ported RAM and 128Kbytes SRAM for firmware download. With C/C++, VB and Delphi self-developing package, high-speed synchronous communication programming is just a breeze. Excellent hardware components and software techniques make C502 perfect for high throughput front-end processing applications.

Designed for high-speed synchronous communication, MOXA C502 is suitable for IBM PC/AT and compatible systems under Windows NT environment.

1.2 Specifications

- On board RISC CPU
- 1M bytes dual port RAM buffer
- 128K bytes SRAM
- Baud rate up to 4Mbps for V.35, 128Kbps for RS-232
- Cable selection V.35/RS-232 interface compatible
- Free Windows NT 4.0 developing tool
- High performance SCA HD64570-10 serial communication adapter with DMA controller
- IRQ:2,3,4,5,7,9,10,11,12,15(jumper selectable)
- System: PC ISA/EISA bus

1.3 Packaging List

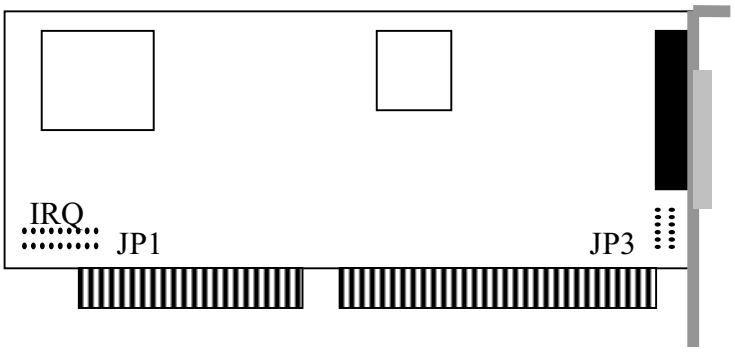
Upon unpacking MOXA C502, you will find the following items:

- MOXA C502 Sync Board
- RS-232 or V.35 Connection cable
- MOXA C502 user's manual.
- MOXA C502 driver diskette for Windows NT

2
Hardware Installation

1. Power off PC and remove PC cover.

2. Configure C502
   - ☐ IRQ number: Find an available IRQ number in your system and setup jumper JP1. There are 9 IRQ numbers you can choose from. **If you are installing more than one C502 card, their IRQ numbers must be set the same.**
   - ☐ Base address: Choose a base address (occupying 16KB) which is not used by expansion memory or other add-on cards. There are 6 memory bank you can choose from at jumper JP3. **If you are installing more than one C502 board, each board must have a unique address.**



Warning: Make sure your system is powered off before you start installing the I/O card. If you don't, you may risk damaging both your system and the card.

3. After the setting has been done, choose an available 16-bit expansion slot. Remove the retaining screw and put it aside.

4. Remove the slot cover.

5. Orient C502 edge connector facing downward. Place it in the I/O slot. Press the card firmly into the plastic edge connector socket on the computer motherboard.

6. Use retaining screw to secure C502 to the rear panel. You can install up to four C502 cards in your system at one time.

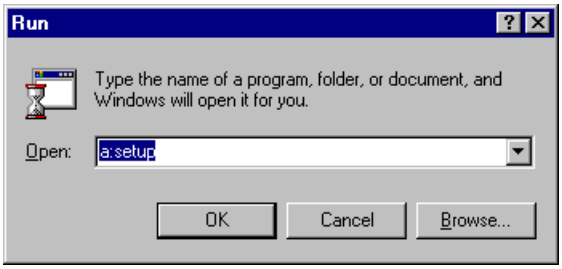
7. Put back PC cover.

3 Software Installation

C502 software includes Windows NT driver, Configuration, Win32 API, and uninstallation program.

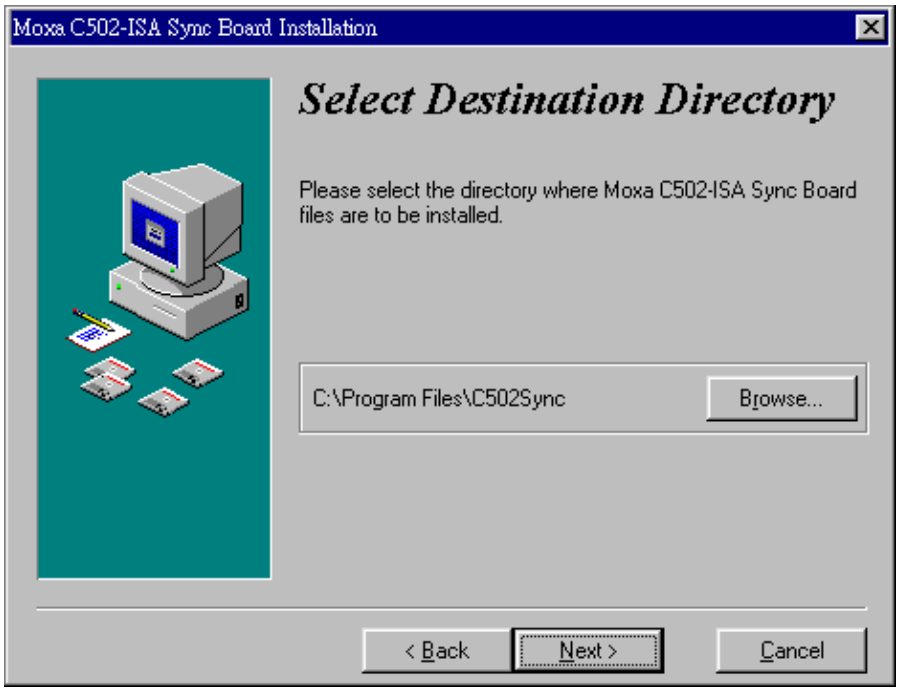
3.1 Install C502 Windows NT Driver

1. Insert C502 Driver for Windows NT disk into drive A. From “Start” menu, click “Run”.
2. Type "a:\setup.exe", then click OK to continue.



3. Setup program prompts you a welcome message and asks if you want to install C502 program now. Click “Next” to continue.

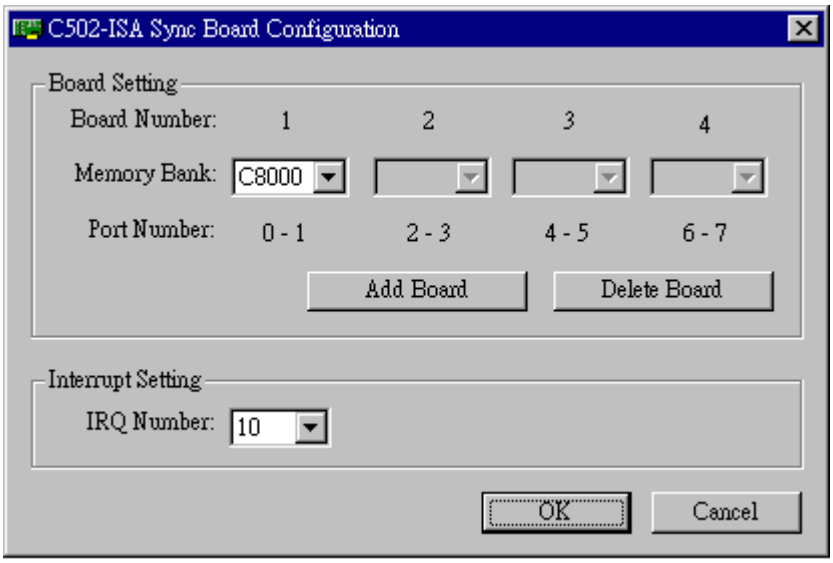
4. Enter the name of directory to install the C502 files. You may click “Next” to use default directory name.



5. Configuration program will start after installation completes.

3.2 C502 Configuration

1. From ‘Start’ menu, select ‘Program’→ ‘Moxa Sync Board’→‘Configuration’
2. Press ‘Add Board’ button for board configuration. Max. 4 boards can be installed in one system.



3. Each C502 board must have a unique memory bank address. Enter the value you set on Jumper 3 while configuring C502 board.
4. Choose the IRQ number you set on C502 board. The IRQ number is shared by every C502 on board.
5. If you want remove one board, press ‘Delete Board’. Board deleting sequence is board 4, board 3, board 2, and board 1.
6. You must set at least one board.

3.3 Remove C502 Windows NT Driver

From ‘Start’ menu, select ‘Program’→‘Moxa Sync Board’→‘Uninstall’. It will automatically remove all C502 programs.

4
API Programming Library

4.1 API Programming Library Notes

MOXA C502 supports C/C++, VB and Delphi language. If you use VB, include ‘syncapi.bas’ file in your project. If you use Delphi, include ‘syncapi.pas’. All of the languages need ‘syncapi.dll’ file, which is copied to your PC when you install C502 driver.

The ‘syncapi.lib’ library file is used for Microsoft C/C++. If you’re using Borland C/C++ Compiler, please use the utility ‘implib.exe’ of Borland C++ to execute "**implib -c syncapib.lib syncapi.dll**" and obtain Borland-compliant library file ‘syncapib.lib’ from the dynamic link library "syncapi.dll".

MOXA C502 supports block/non-block mode for read/write function with your application.

Following is the return code list you may encounter when calling these library functions:

Return Code	Output	Description
SYIO_OK	0	function OK
SYIO_BADPORT	-1	no such port or not opened
SYIO_OPENED	-2	port opened
SYIO_BADPARM	-3	parameter error
SYIO_WIN32FAIL	-4	call win32 function fail, call GetLastError() function to get the return code
SYIO_ABORT	-5	abort writing
SYIO_TIMEOUT	-6	read or write timeout
SYIO_BUFFERTOOSHORT	-8	buffer too short

4.2 Library Function Description

1. syio_Open

Description:

Open one port and set port to default value. Port default value is Tx clock out, baud rate 38400, CRC CCITT_1, and data encoding NRZ.

Syntax:

C/C++

```
int WINAPI syio_Open(int port);
Input     : int port (port number 0 ~ 7)
Output    : refer to return code list
```

VB

```
Declare Function syio_Open Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

```
function syio_Open(port: Longint): Longint; stdcall;
implementation
function syio_Open; external ‘syncapi.dll’;
```

2. syio_Close

Description:

Close one opened port. If there is no need to use one port, you can call this function. It will wait the data to send over. If there is no data to send in 3 seconds, it flush output and input data on the buffer of the driver.

Syntax:

C/C++

```
int WINAPI syio_Close(int port);
Input     : int port (port number 0 ~ 7)
Output    : refer to return code list
```

VB

```
Declare Function syio_Close Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

function syio_Close(port: Longint): Longint; stdcall;

implementation

function syio_Close; external 'syncapi.dll';

3. syio_Write

Description:

Send data. If you set write-timeout to zero, it will write the data to dual-port DRAM on board and return as soon as possible. If you set write-timeout to specific value, it will block syio_Write function call until data writing is completed or times out.

Syntax:

C/C++

int WINAPI syio_Write(int port, char *buf, int len);

Input:	int port	:port number 0 ~ 7
	char *buf	: to-send data buffer pointer
	Int len	: to-send data length
Output	>= 0	: sent data length
	< 0	: refer to return code list

VB

Declare Function syio_Write Lib "syncapi.dll" (ByVal port As Long, ByRef buf As Byte, ByVal len As Long) As Long

Delphi

function syio_Write(port: Longint; buf: PChar; len: Longint): Longint; stdcall;

implementation

function syio_Write; external 'syncapi.dll';

4. syio_Read

Description:

Receive data from remote device. If you set the read-timeout to zero, it will return as soon as possible when there is no incoming data. If you set read-timeout to non-zero value, it will block syio_Read function call until data reading is over or times out.

Syntax:

C/C++

int WINAPI syio_Read(int port, char *buf, int len);

Input:	int port	:port number 0 ~ 7
	char *buf	: to-receive data buffer pointer
	int len	: to-receive data length
Output	>= 0	: receiving data length
	< 0	: refer to return code list

VB

Declare Function syio_Read Lib “syncapi.dll” (ByVal port As Long, ByRef buf As Byte, ByVal len As Long) As Long

Delphi

function syio_Read(port: Longint; buf: PChar; len: Longint): Longint; stdcall;

implementation

function syio_Read; external ‘syncapi.dll’;

5. syio_Flush

Description:

Flush received or to-be-send data on the driver.

Syntax:

C/C++

int WINAPI syio_Flush(int port, int mode);

Input:	int port	:port number 0 ~ 7
	int mode	:FLUSH_INPUT, FLUSH_OUTPUT or FLUSH_ALL
Output	: refer to return code list	

VB

Declare Function syio_Flush Lib “syncapi.dll” (ByVal port As Long, ByVal mode As Long) As Long

Delphi

function syio_Flush(port, mode: Longint): Longint; stdcall;

implementation

function syio_Flush; external ‘syncapi.dll’;

6. syio_View

Description:

Preview data. It functions like syio_Read, but data stays on the driver afterward. It has not timeout value.

Syntax:

C/C++

int WINAPI syio_View(int port, char*buf, int len);

Input:	int port	:port number 0 ~ 7
	char *buf	: to-view data buffer pointer
	int len	: to-view data length
Output	>= 0	: viewing data length
	< 0	: refer to return code list

VB

Declare Function syio_View Lib “syncapi.dll” (ByVal port As Long, ByRef buf As Byte, ByVal len As Long) As Long

Delphi

function syio_View(port: Longint; buf: PChar; len: Longint): Longint; stdcall;

implementation

function syio_View; external ‘syncapi.dll’;

7. syio_SetBaud

Description:

Set baud rate. Baud rate setting is invalid if Tx Clock is set ‘in’. You set Tx clock ‘out’ to activate baud rate setting.

Syntax:

C/C++

int WINAPI syio_SetBaud(int port, int speed);

Input:	int port	:port number 0 ~ 7
	int speed	: to-set baud rate
Output:	refer to return code list	

VB

Declare Function syio_SetBaud Lib “syncapi.dll” (ByVal port As Long, ByVal speed As Long) As Long

Delphi

function syio_SetBaud(port, speed: Longint): Longint; stdcall;

implementation

function syio_SetBaud; external 'syncapi.dll';

8. syio_GetBaud

Description:

Get baud rate setting value.

Syntax:

C/C++

int WINAPI syio_GetBaud(int port);

Input:	int port	: port number 0 ~ 7
	int speed	: set baud rate
Output	>= 0	: set baud rate
	< 0	: refer to return code list

VB

Declare Function syio_GetBaud Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

function syio_GetBaud(port: Longint): Longint; stdcall;

implementation

function syio_GetBaud; external 'syncapi.dll';

9. syio_SetReadTimeouts

Description:

Set the syio_Read timeout value. Please refer to syio_Read function.

Syntax:

C/C++

int WINAPI syio_SetReadTimeouts(int port, DWORD timeouts);

Input:	int port	:port number 0 ~ 7
	DWORD timeouts	: to-set timeouts value. Time unit is millisecond
Output:	refer to return code list	

VB

Declare Function syio_SetReadTimeouts Lib “syncapi.dll” (ByVal port As Long, ByVal timeouts As Long) As Long

Delphi

function syio_SetReadTimeouts(port, timeouts: Longint): Longint; stdcall;

implementation

function syio_SetReadTimeouts; external ‘syncapi.dll’;

10. syio_GetReadTimeouts

Description:

Get read-timeout setting value. Please refer to the syio_Read and syio_SetReadTimeouts function.

Syntax:

C/C++

int WINAPI syio_GetReadTimeouts(int port, DWORD *timeouts);

Input:	int port	:port number 0 ~ 7
	DWORD *timeouts	: to get timeouts pointer.
Output:	refer to return code list	

VB

Declare Function syio_GetReadTimeouts Lib “syncapi.dll” (ByVal port As Long, ByRef timeouts As Long) As Long

Delphi

function syio_GetRedTimeouts(port: Longint; var timeouts: Longint): Longint; stdcall;

implementation

function syio_GetReadTimeouts; external ‘syncapi.dll’;

11. syio_SetWriteTimeouts

Description:

Set write-timeout setting value. Please refer to the syio_Write function.

Syntax:

C/C++

int WINAPI syio_SetWriteTimeouts(int port, DWORD timeouts);

Input:	int port	:port number 0 ~ 7
	DWORD timeouts	: to set write timeouts value
Output:	refer to return code list	

VB

Declare Function syio_SetWriteTimeouts Lib “syncapi.dll” (ByVal port As Long, ByVal timeouts As Long) As Long

Delphi

function syio_SetWriteTimeouts(port, timeouts: Longint): Longint; stdcall;

implementation

function syio_SetWriteTimeouts; external ‘syncapi.dll’;

12. syio_GetWriteTimeouts

Description:

Get write-timeout setting value. Please refer to syio_Write and syio_SetWriteTimeouts function.

Syntax:

C/C++

int WINAPI syio_GetWriteTimeouts(int port, DWORD *timeouts);

Input:	int port	:port number 0 ~ 7
	DWORD *timeouts	: to get timeouts pointer.
Output:	refer to return code list	

VB

Declare Function syio_GetWriteTimeouts Lib “syncapi.dll” (ByVal port As Long, ByRef timeouts As Long) As Long

Delphi

function syio_GetWriteTimeouts(port: Longint; var timeouts: Longint): Longint; stdcall;

implementation

function syio_GetWriteTimeouts; external ‘syncapi.dll’;

13. syio_AbortRead

Description:

Abort the blocked syio_Read function call.

Syntax:

C/C++

int WINAPI syio_AbortRead(int port);

Input: int port :port number 0 ~ 7
Output: refer to return code list

VB

Declare Function syio_AbortRead Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_AbortRead(port: Longint): Longint; stdcall;

implementation

function syio_AbortRead; external ‘syncapi.dll’;

14. syio_AbortWrite

Description:

Abort the blocked syio_Write function call.

Syntax:

C/C++

int WINAPI syio_AbortWrite(int port);

Input: int port :port number 0 ~ 7
Output: refer to return code list

VB

Declare Function syio_AbortWrite Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_AbortWrite(port: Longint): Longint; stdcall;

implementation

function syio_AbortWrite; external ‘syncapi.dll’;

15. syio_DTR

Description:

Set DTR pin on or off.

Syntax:

C/C++

int WINAPI syio_DTR(int port, int mode);

Input:	int port	:port number 0 ~ 7
	int mode	: 0 for off, 1 for on
Output:	refer to return code list	

VB

Declare Function syio_DTR Lib “syncapi.dll” (ByVal port As Long, ByVal mode As Long) As Long

Delphi

function syio_DTR(port, mode: Longint): Longint; stdcall;

implementation

function syio_DTR; external ‘syncapi.dll’;

16. syio_RTS

Description:

Set RTS pin on or off.

Syntax:

C/C++

int WINAPI syio_RTS(int port, int mode);

Input:	int port	:port number 0 ~ 7
	int mode	: 0 for off, 1 for on
Output:	refer to return code list	

VB

Declare Function syio_RTS Lib “syncapi.dll” (ByVal port As Long, ByVal mode As Long) As Long

Delphi

function syio_RTS(port, mode: Longint): Longint; stdcall;

implementation

function syio_RTS; external 'syncapi.dll';

17. syio_SkipFrame

Description:

Skip first received frame on the buffer of the driver. The skipped frame will be aborted and not be read by the application.

Syntax:

C/C++

int WINAPI syio_SkipFrame(int port);

Input:	int port	:port number 0 ~ 7
Output:	refer to return code list	

VB

Declare Function syio_SkipFrame Lib "syncapi.dll" (ByVal port As Long)

Delphi

function syio_SkipFrame(port: Longint): Longint; stdcall;

implementation

function syio_SkipFrame; external 'syncapi.dll';

18. syio_InFrame

Description:

Get the number of received frames on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_InFrame(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: received frames
	< 0	: refer to return code list

VB

Declare Function syio_InFrame Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

function syio_InFrame(port: Longint): Longint; stdcall;

implementation

function syio_InFrame; external ‘syncapi.dll’;

19. syio_OutFrame

Description:

Get the number of to-be-send frames on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_OutFrame(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: to-send frames
	< 0	: refer to return code list

VB

Declare Function syio_OutFrame Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_OutFrame(port: Longint): Longint; stdcall;

implementation

function syio_OutFrame; external ‘syncapi.dll’;

20. syio_InFreeFrame

Description:

Get the number of free input frames on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_InFreeFrame(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: input free frames
	< 0	: refer to return code list

VB

Declare Function syio_InFreeFrame Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_InFreeFrame(port: Longint): Longint; stdcall;

implementation

function syio_InFreeFrame; external 'syncapi.dll';

21. syio_OutFreeFrame

Description:

Get the number of free output frames on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_OutFreeFrame(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: output free frames
	< 0	: refer to return code list

VB

Declare Function syio_OutFreeFrame Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

function syio_OutFreeFrame(port: Longint): Longint; stdcall;

implementation

function syio_OutFreeFrame; external 'syncapi.dll';

22. syio_SetDataEncoding

Description:

Set data encoding mode. NRZ and NRZI are supported.

Syntax:

C/C++

int WINAPI syio_SetDataEncoding(int port, int mode);

Input:	int port	:port number 0 ~ 7
	int mode	: NRZ or NRZI
Output:	refer to return code list	

VB

Declare Function syio_SetDataEncoding Lib "syncapi.dll" (ByVal port As Long, ByVal mode As Long) As Long

Delphi

function syio_SetDataEncoding(port, mode: Longint): Longint; stdcall;

implementation

function syio_SetDataEncoding; external 'syncapi.dll';

23. syio_GetDataEncoding

Description:

Get data encoding mode setting value.

Syntax:

C/C++

int WINAPI syio_GetDataEncoding(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: data encoding mode NRZ or NRZI
	< 0	: refer to return code list

VB

Declare Function syio_GetDataEncoding Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

function syio_GetDataEncoding(port: Longint): Longint; stdcall;

implementation

function syio_GetDataEncoding; external 'syncapi.dll';

24. syio_SetCRCMode

Description:

Set CRC mode. CCITT initialized 0, all 1's, or none CRC are supported. HDLC protocol can only use CCITT CRC.

Syntax:

C/C++

int WINAPI syio_SetCRCMode(int port, int mode);

Input:	int port	:port number 0 ~ 7
	int mode	: NONE, CCITT_0 or CCITT_1
Output:	refer to return code list	

VB

Declare Function syio_SetCRCMode Lib “syncapi.dll” (ByVal port As Long, ByVal mode As Long) As Long

Delphi

```
function syio_SetCRCMode(port, mode: Longint): Longint; stdcall;
implementation
function syio_SetCRCMode; external ‘syncapi.dll’;
```

25. syio_GetCRCMode

Description:

Get CRC mode setting value.

Syntax:

C/C++

```
int WINAPI syio_GetCRCMode(int port);
    Input:    int port    : port number 0 ~ 7
    Output    >= 0        : CRC value setting
              < 0         : refer to return code list
```

VB

Declare Function syio_GetCRCMode Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

```
function syio_GetCRCMode(port: Longint): Longint; stdcall;
implementation
function syio_GetCRCMode; external ‘syncapi.dll’;
```

26. syio_LineStatus

Description:

Get line status. The firmware is polling line status every 50ms.

Syntax:

C/C++

int WINAPI syio_LineStatus(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: the line status- bit 0 for DCD, bit 1 for DSR, bit 2 for CTS, bit on(1) for status pin on, bit off(0) for status pin off
	< 0	: refer to return code list

VB

Declare Function syio_LineStatus Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_LineStatus(port: Longint): Longint; stdcall;

implementation

function syio_LineStatus; external ‘syncapi.dll’;

27. syio_InQueue

Description:

Get the received data bytes on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_InQueue(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: received bytes
	< 0	: refer to return code list

VB

Declare Function syio_InQueue Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_InQueue(port: Longint): Longint; stdcall;

implementation

function syio_InQueue; external ‘syncapi.dll’;

28. syio_OutQueue

Description:

Get to-be-sent data bytes on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_OutQueue(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: to-be-sent bytes
	< 0	: refer to return code list

VB

Declare Function syio_OutQueue Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_OutQueue(port: Longint): Longint; stdcall;

implementation

function syio_OutQueue; external ‘syncapi.dll’;

29. syio_InFree

Description:

Get free data bytes space on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_InFree(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: input free bytes
	< 0	: refer to return code list

VB

Declare Function syio_InFree Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_InFree(port: Longint): Longint; stdcall;

implementation

function syio_InFree; external ‘syncapi.dll’;

30. syio_OutFree

Description:

Get free output data bytes space on the buffer of the driver.

Syntax:

C/C++

int WINAPI syio_OutFree(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: output free bytes
	< 0	: refer to return code list

VB

Declare Function syio_OutFree Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_OutFree(port: Longint): Longint; stdcall;

implementation

function syio_OutFree; external ‘syncapi.dll’;

31. syio_FrameIrq

Description:

Set the event ‘number of received frame’. You can specify a function to be called when frame event happens. If the function is set NULL, frame event will be cleared.

Syntax:

C/C++

int WINAPI syio_FrameIrq(int port, VOID (CALLBACK *func)(int port), int framecnt);

Input:	int port	:port number 0 ~ 7
	VOID (CALLBACK *func)(int port)	: the function to be called when this event happens
	int framecnt	:Number of received frames to call the function. It must be greater than zero
Output:	refer to return code list	

VB

Declare Function syio_FrameIrq Lib “syncapi.dll” (ByVal port As Long, ByVal func As Long, ByVal framecnt As Long) As Long

Delphi

```
Type
IrqProc1 = procedure(port: Longint); stdcall;
function syio_FrameIrq(port: Longint; func: IrqProc1; framecnt: Longint): Longint;
stdcall;
implementation
function syio_FrameIrq; external ‘syncapi.dll’;
```

32. syio_ModemIrq

Description:

Set the event ‘modem status change ’. You can specify a function to be called when modem CTS, DCD, DSR on/off status changes. If the function is set NULL, modem event will be cleared.

Syntax:

C/C++

```
int WINAPI syio_ModemIrq(int port, VOID (CALLBACK *func)(int port, int status),
int mode);
```

Input:	int port	:port number 0 ~ 7
	VOID (CALLBACK *func)(int port, int status)	: the function to be called when this event happens
	int mode	:Types of modem status change At last one modem status has to be set.
Output:	refer to return code list	

VB

Declare Function syio_ModemIrq Lib “syncapi.dll” (ByVal port As Long, ByVal func As Long, ByVal mode As Long) As Long

Delphi

type

IrqProc2 = procedure(port, status: Longint); stdcall;

function syio_ModemIrq(port: Longint; func: IrqProc2; mode: Longint): Longint;

stdcall;

implementation

function syio_ModemIrq; external 'syncapi.dll';

33. syio_TxEmptyIrq

Description:

Set the event 'Tx Empty'. You can specify a function to be called when Tx Empty event happens. If the function is set NULL, Tx Empty event will be cleared.

Syntax:

C/C++

int WINAPI syio_TxEmptyIrq(int port, VOID (CALLBACK *func)(int port);

Input:	int port	:port number 0 ~ 7
	VOID (CALLBACK *func)(int port)	: the function to be called when this event happens
Output:	refer to return code list	

VB

Declare Function syio_TxEmptyIrq Lib "syncapi.dll" (ByVal port As Long, ByVal func As Long) As Long

Delphi

type

IrqProc1 = procedure(port: Longint); stdcall;

function syio_TxEmptyIrq(port: Longint; func: IrqProc1): Longint; stdcall;

implementation

function syio_TxEmptyIrq; external 'syncapi.dll';

34. syio_SetTxClockDir

Description:

Set Tx clock direction ‘in’ or ‘out’. Tx clock ‘in’ uses different pin on connector from clock ‘out’.

Syntax:

C/C++

int WINAPI syio_SetTxClockDir(int port, int direction);

Input:	int port	:port number 0 ~ 7
	int direction	:IN or OUT
Output:	refer to return code list	

VB

Declare Function syio_SetTxClockDir Lib “syncapi.dll” (ByVal port As Long, ByVal direction As Long) As Long

Delphi

function syio_SetTxClockDir(port, direction: Longint): Longint; stdcall;

implementation

function syio_SetTxClockDir; external ‘syncapi.dll’;

35. syio_GetTxClockDir

Description:

Get Tx clock direction setting value.

Syntax:

C/C++

int WINAPI syio_GetTxClockDir(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: clock direction
	< 0	: refer to return code list

VB

Declare Function syio_GetTxClockDir Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_GetTxClockDir(port: Longint): Longint; stdcall;

implementation

function syio_GetTxClockDir; external 'syncapi.dll';

36. syio_TxDisable

Description:

Disable Tx transmission.

Syntax:

C/C++

int WINAPI syio_TxDisable(int port);

Input: int port :port number 0 ~ 7
Output: refer to return code list

VB

Declare Function syio_TxDisable Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

function syio_TxDisable(port: Longint): Longint; stdcall;

implementation

function syio_TxDisable; external 'syncapi.dll';

37. syio_TxEnable

Description:

Enable transmission halted by syio_TxDisable.

Syntax:

C/C++

int WINAPI syio_TxEnable(int port);

Input: int port :port number 0 ~ 7
Output: refer to return code list

VB

Declare Function syio_TxEnable Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

```
function syio_TxEnable(port: Longint): Longint; stdcall;
implementation
function syio_TxEnable; external 'syncapi.dll';
```

38. syio_TxStatus

Description:

Get Tx status, 'disable' or 'enable'.

Syntax:

C/C++

```
int WINAPI syio_TxStatus(int port);
    Input:   int port    : port number 0 ~ 7
    Output   >= 0        : Tx status, 0 for disable, 1 for enable
             < 0         : refer to return code list
```

VB

```
Declare Function syio_TxStatus Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi

```
function syio_TxStatus(port: Longint): Longint; stdcall;
implementation
function syio_TxStatus; external 'syncapi.dll';
```

39. syio_GetFirstFrameLen

Description:

Get first received frame length.

Syntax:

C/C++

```
int WINAPI syio_GetFirstFrameLen(int port);
    Input:   int port    : port number 0 ~ 7
    Output   >= 0        : the first frame length
             < 0         : refer to return code list
```

VB

Declare Function syio_GetFirstFrameLen Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_getFirstFrameLen(port: Longint): Longint; stdcall;

implementation

function syio_GetFirstFrameLen; external ‘syncapi.dll’;

40. syio_GetBoardID

Description:

Get board ID number. Default number is 1. Other ID numbers are available for OEM user.

Syntax:

C/C++

int WINAPI syio_GetBoardID(int port);

Input:	int port	: port number 0 ~ 7
Output	>= 0	: 1 only
	< 0	: refer to return code list

VB

Declare Function syio_GetBoardID Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_GetBoardID(port: Longint): Longint; stdcall;

implementation

function syio_GetBoardID; external ‘syncapi.dll’;

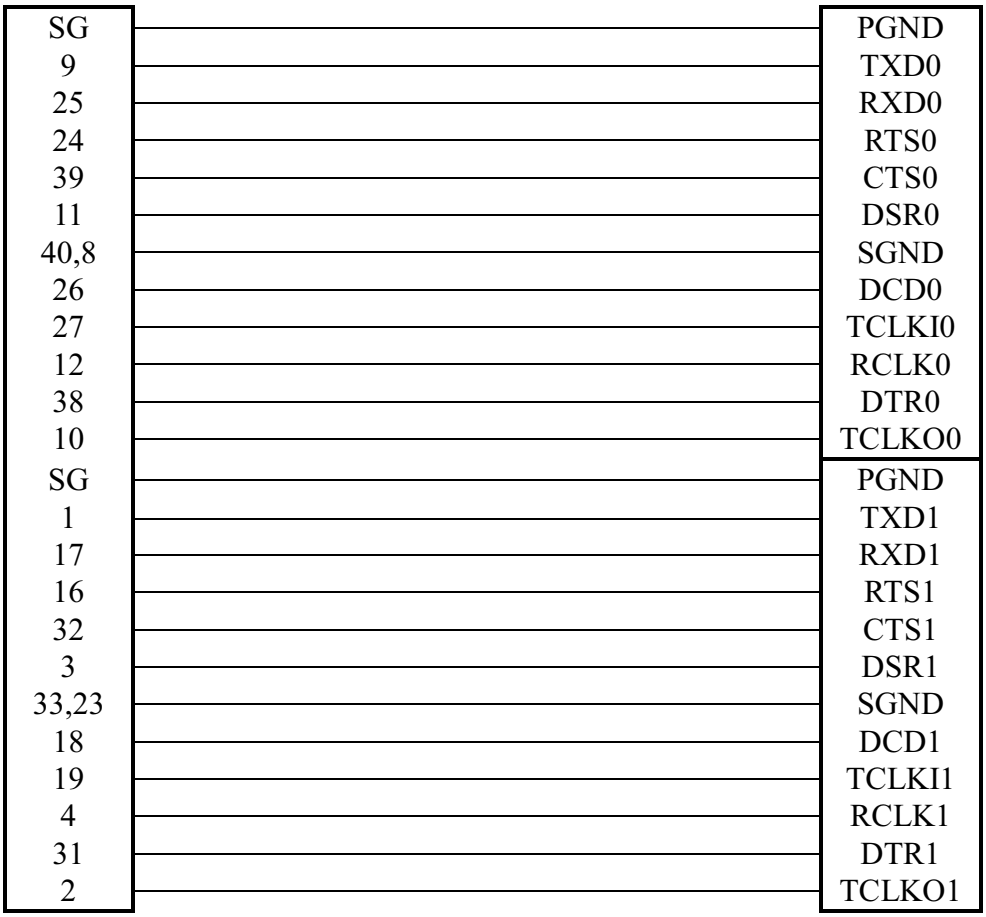
A

Appendix A RS232/V.24 Connection

RS-232/V.24 Pin Assignment for MOXA C502

Pin #	Signal	Name	Direction	CCITT #
2	TXD	Transmit Data	Output	103
3	RXD	Receive Data	Input	104
4	RTS	Request to Send	Output	105
5	CTS	Clear to Send	Input	106
6	DSR	Data Set Ready	Input	107
7	SGND	Signal Ground	Common	102
8	DCD	Data Carrier Detect	Input	109
15	TCLKI	Transmit Clock	Input	114
17	RCLK	Receive Clock	Input	115
20	DTR	Data Terminal Ready	Output	108
24	TCLKO	Transmit Clock	Output	113

RS-232 Dual-Port DB44 Cable Connections (Port 0 and Port 1)



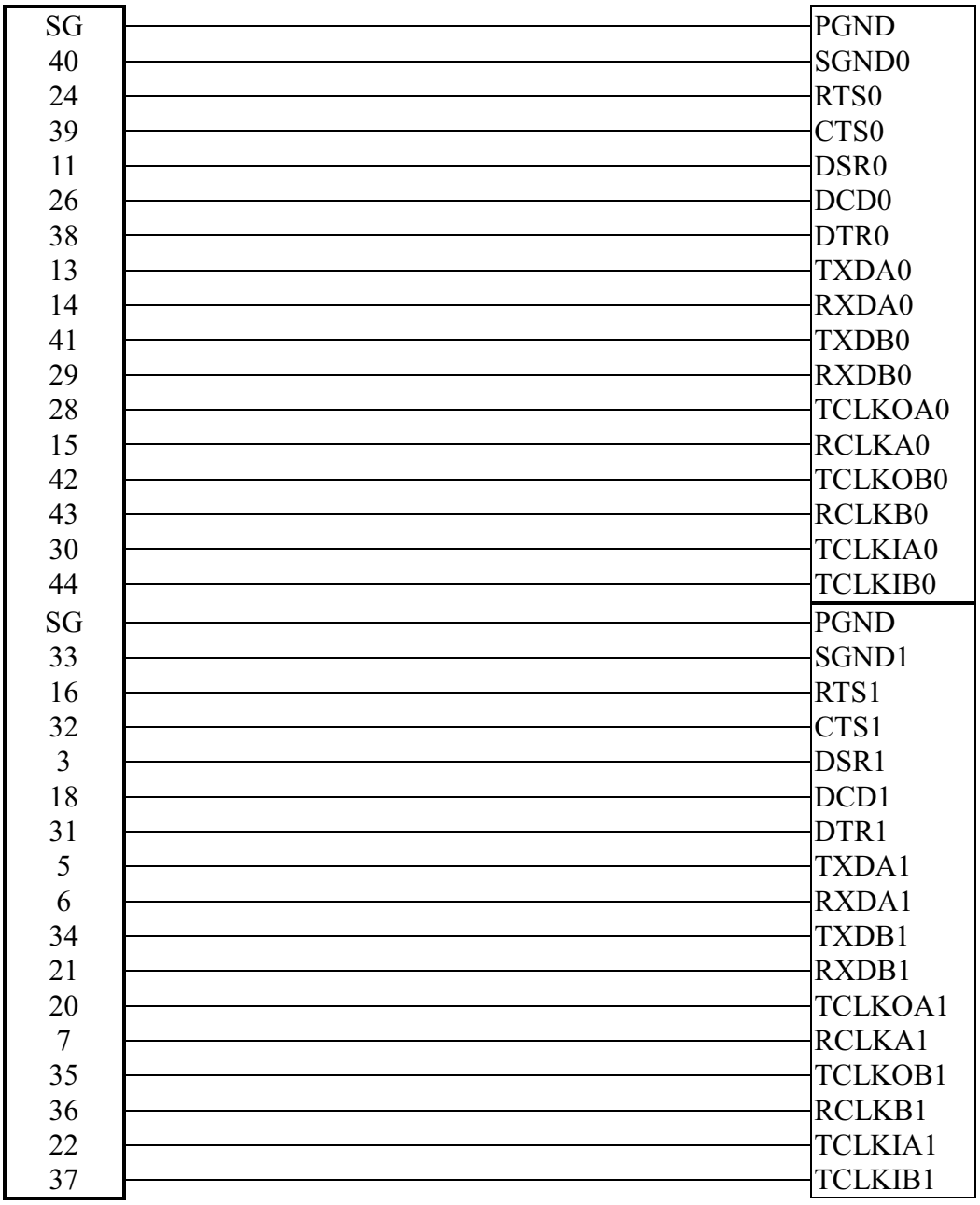
B

Appendix B V.35 Connection

V.35 Pin Assignment for MOXA C502

Pin #	Signal	Name	Direction	CCITT #
A	PGND	Protective Ground	Common	101
B	SGND	Signal Ground	Common	102
C	RTS	Request to Send	Output	105
D	CTS	Clear to Send	Input	106
E	DSR	Data Set Ready	Input	107
F	DCD	Data Carrier Detect	Input	109
H	DTR	Data Terminal Ready	Output	108
P	TXDA	Transmit Data	Output	103A
R	RXDA	Receive Data	Input	104A
S	TXDB	Transmit Data	Output	103B
T	RXDB	Receive Data	Input	104B
U	TCLKOA	Transmit Clock(DTE)	Output	113A
V	RCLKA	Receive Clock(DCE)	Input	115A
W	TCLKOB	Transmit Clock(DTE)	Output	113B
X	RCLKB	Receive Clock(DCE)	Input	115B
Y	TCLKIA	Transmit Clock(DCE)	Input	114A
AA	TCLKIB	Transmit Clock(DCE)	Input	114B

V.35 Dual-Port DB44 Cable Connections (Port 0 and Port 1)



C

Appendix C Trouble Shooting

1. Download BIOS or firmware file fails

   Possible problem types and solutions:

   a) C502 base address conflicts with the BIOS ROM Shadow. Disable the BIOS ROM Shadow C502 uses. For example, if you set C502 to base address C8000 (or C800:0000), then C800:0000 ROM Shadow must be disabled.

   b) C502 base address conflicts with that of other interface cards such as SCSI or LAN cards. Adjust the address to forestall the conflict.

   c) C502 is not properly plugged in a 16-bit slot. Reinstall C502 and make sure it fits well this time.

   d) C502 does not function well. Kindly return for repair.

2. C502 driver initializes OK but can not transfer any data.

   Check if wrong cable wiring. Refer to Appendix for precise pin assignment of communication port and its cable wiring. To be sure the transmit clock direction is OK.

Problem Report Form

C502 Dual-Port Sync Board

Customer name:	
Company:	
Tel:	Fax:
Email:	Date:

1. **Moxa Product**: ☐ C502-ISA/RS232 ☐ C502-ISA/V.35
2. **Serial Number**:_____(Please see the rear panel of the board)
3. **Driver Version**: _____
4. **Problem Description**: Please describe the symptom as clear as possible including the return message you see. We may have to follow your description to reproduce the symptom.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

RETURN PROCEDURE

For product repair, exchange or refund, the customer must:

- Provide evidence of original purchase
- Obtain a Product Return Agreement (PRA) from the sales representative or dealer
- Fill out the Problem Report Form (PRF) as detailed as possible for shorter product repair time.
- Carefully pack the product in anti-static package, and send it, pre-paid, to the dealer. The PRA should show on the outside of the package, and include a description of the problem along with the return address and telephone number of a technical contact.